Data and Application Security in Modern Scenarios

Stefano Paraboschi
Università degli Studi di Bergamo

Overall technological trends

- Fast increase of:
  - Storage
  - Long-distance bandwidth
  - Computation
- Slower increase of:
  - CPU/RAM interface
  - CPU/Storage interface
  - LAN bandwidth
  - Network latency
- Reduced costs
  - Multiple devices per user
- Wireless networks
  - Mobile devices always connected
- New element (briefly discussed at the end): ML/AI

Impact of overall trends on system design

- Decrease of storage cost and corresponding increase of information value
- Increase of the convenience of remote storage
  - Reliable, inexpensive, accessible
  - Local storage only for caching
- Increase of the need of sharing resources, due to the larger number of devices and their portability

- Examples of the impact:
  - Shift from licenses to own to licenses to use content
  - Access to music/video content by streaming

Impact on security

- In the business world, originally attention was mostly dedicated to network security and authentication
  - Goal: separate insiders (known, trusted, well-behaving) from outsiders (unknown, untrusted, misbehaving)
- The presence of multiple devices and the need to support access to remotely stored data removes clear boundaries
  - the assumption of a clear separation based on network configuration or authentication does not hold anymore
- Smartphones show a clear revision of classical approaches
  - Adaptation of multi-user protection to isolate apps
  - Complete erasure of local data when resetting the system

Security focus shifts away from computation

- Low cost and convenient remote storage lead to a model of disposable computing devices
- The device offers a clean environment to operate over data, with strict isolation
- Within a user domain, separate execution environments, each associated with tasks that receive distinct privileges to access data
  - Virtualization, containers, etc.
- It resembles security of Web browsers, with a more structured representation of data

Secure execution environments

- There are many complex system issues that need to be solved
- It is still important to support the realization of more secure execution environments
  - E.g., support for JS and WASM execution environments
- But the main threat deriving from insecure execution environments is the compromise of systems and the leakage of sensitive data

M. Abbadini, D. Facchinetti, G. Oldani, M. Rossi, S. Paraboschi, “Cage4Deno: A Fine-Grained Sandbox for Deno Subprocesses”, in ACM ASIACCS 2023.

M. Abbadini, D. Facchinetti, G. Oldani, M. Rossi, S. Paraboschi, “NatiSand: Native Code Sandboxing for JavaScript Runtimes”, RAID 2023.

Security focus moving toward data

- Data protection and flexible access policies promise to assume a greater role

Data protection in modern systems

Several dimensions come into play when considering the protection of data in current scenarios

- Security properties
- Cloud architecture
- Access mode
- Access policy granularity

Dimension: Classical security properties

- Confidentiality
  - Encryption, with keys unavailable to the server (*honest-but-curious* model)
- Integrity
  - Data collection is large and dynamic (on value and location); probabilistic methods become more interesting
- Availability
  - Checks on properties that are not falsifiable (e.g., network latency)
  - SLAs have to be automatically supported

Dimension: Cloud architecture

- Single client/Single provider
  - Both client and provider use multiple machines, but each refers to a distinct single trust domain
- Multi-client/*-Provider
  - Protected sharing of resources must be supported, with no violation of security by the provider (protecting the policy itself)
- *-Client/Multi-Provider
  - clouds can be external or internal (hybrid clouds)
  - cloud providers can present a specific focus: storage services or computational services

Dimension: Access mode

- Long-term storage/bulk read-write
  - Simple symmetric encryption is sufficient
- Queries
  - Secure indexes have to be introduced
- Updates
  - Inferences are harder to control

Dimension: Access policy granularity

- Protection of single instance/attribute
  - Each element is classified by the policy, for each user, as accessible or not
- Protection of associations
  - The information to be protected is the association among values, not their plaintext
- Protection of associations on data under an overall privacy constraint
  - k-anonymity, differential privacy
  - security/utility, instead of classical security/performance

Orthogonality of dimensions

- Each combination of dimensions corresponds to a specific scenario
- E.g., use of multiple providers and classical security properties
  - Confidentiality: splitting data, no single provider controls everything
  - Integrity: small amount of redundancy can detect inconsistencies
  - Availability: redundancy has to be introduced in the data, in conflict with confidentiality
  - All the techniques have to consider what happens when providers collude: collusion must not lead to a full compromise

Data protection in modern systems

- The consideration of all combinations creates a large number of scenarios
- A few of the investigated topics will be succinctly described next
  - Selective encryption and Over-encryption
  - Mix&Slice
  - Sentinels and twins

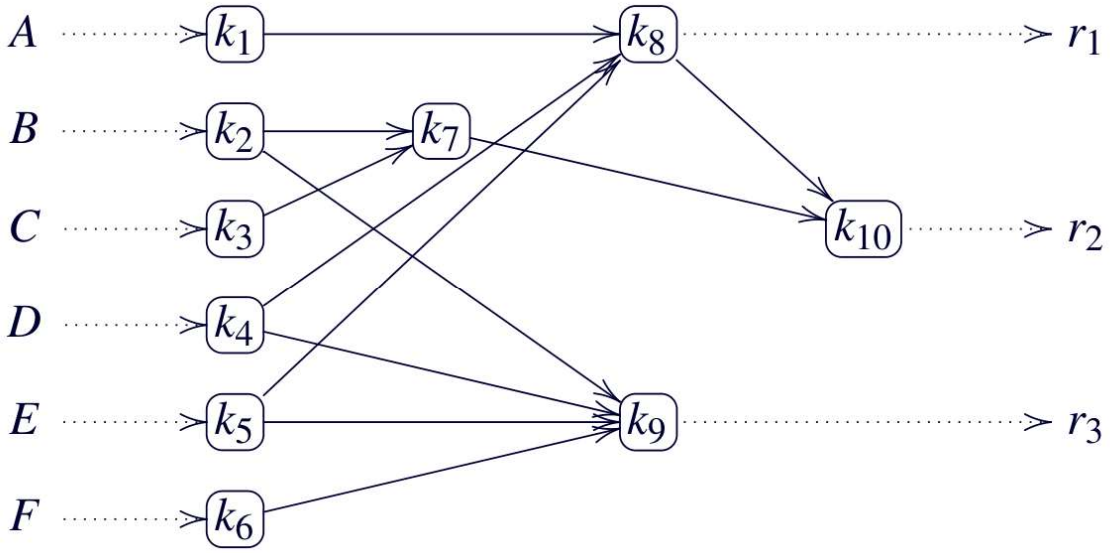
Selective encryption

- Encryption is increasingly used to protect data stored in the Cloud
- In the presence of multiple users, the access policy maps to the ability of each authorized user to get/derive the encryption key for all the resources she should be allowed to access

Basic idea:

- different ACLs imply different encryption keys
- key derivation method limits the number of keys
  - via public tokens a user can derive all keys of the resources she is allowed to access

Selective encryption – Example



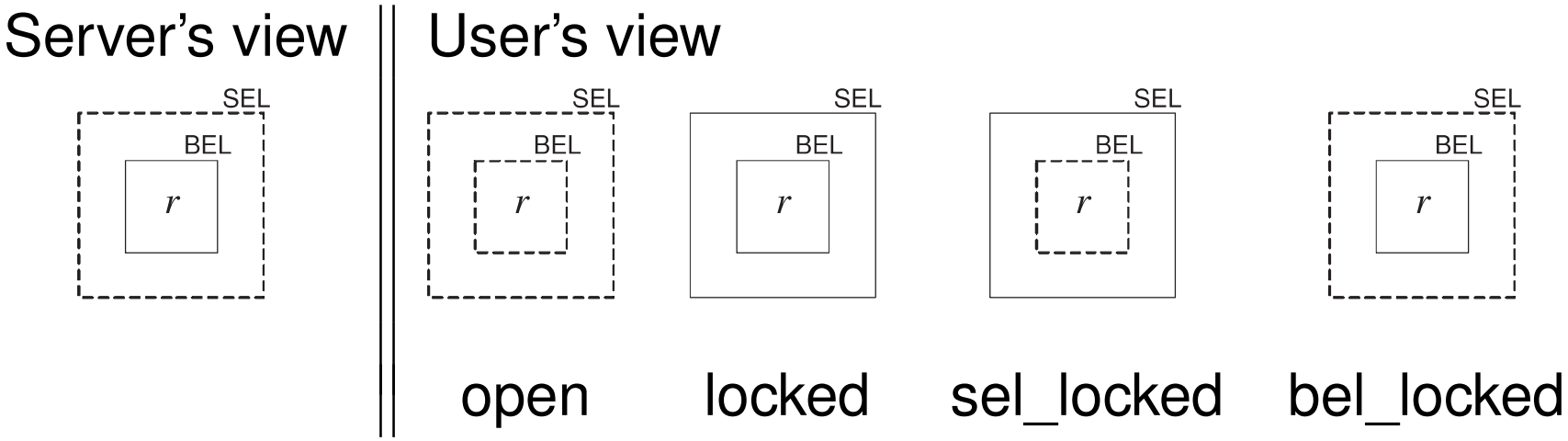
- user A can access $\{r_1, r_2\}$
- user B can access $\{r_2, r_3\}$
- user C can access $\{r_2\}$
- user D can access $\{r_1, r_2, r_3\}$
- user E can access $\{r_1, r_2, r_3\}$
- user F can access $\{r_3\}$

User revocation

Alternatives to manage user revocation?

- The key becomes unavailable to the revoked user
  - efficient, but the user may have kept a local copy of the encryption key
- The resource is re-encrypted with a fresh key
  - inefficient, because the resources have to be read, decrypted, re-encrypted, and written back
    - especially inefficient in a cloud environment
- The resource is protected with an additional encryption layer (*over-encryption*)

Over-encryption



- Each layer is depicted as a fence
  - discontinuous, if the key is known
  - continuous, if the key is not known (protection cannot be passed)

Alternative for user revocation: Mix&Slice

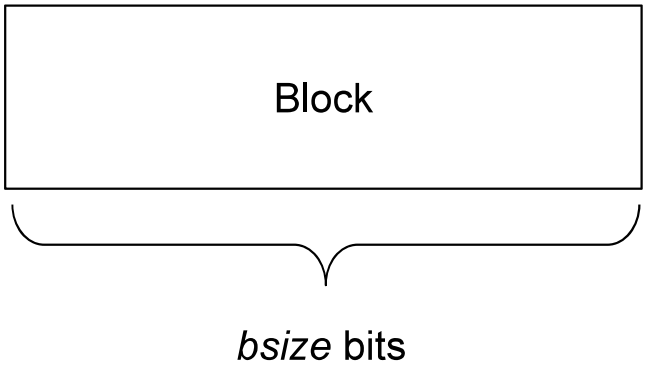
- Use an encryption structure that splits each resource and then mixes all pieces within each portion, making the decryption dependent on the availability of all the pieces.
  - Same principle as All-Or-Nothing-Transform (AONT)
- When the policy changes, update the encrypted representation of a randomly chosen piece.
- If there are n pieces, the resource can become unavailable with the update of $1/n$-th of the resource
  - \+ Protects against revoked users keeping their keys
  - \+ Efficient (arbitrarily more efficient than resource re-encryption)
  - \+ No need of dedicated support by the storage server
- Some AONT approaches do not provide protection in this scenario

---

E. Bacis, S. De Capitani di Vimercati, S. Foresti and S. Paraboschi, M. Rosa, P. Samarati, "Mix&slice for Efficient Access Revocation on Outsourced Data", in IEEE TDSC, 21:3, May/June 2024

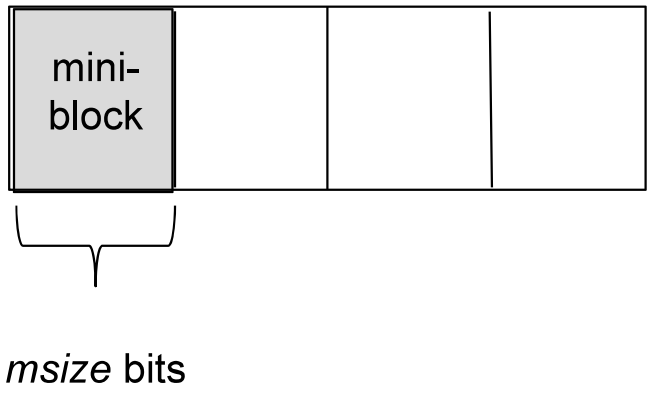
Block

- The block is the unit of work for a symmetric block cipher
- The algorithm used determines the size of the block (*bsize* bits)
- AES is today the most common option (by far) $\rightarrow$ *bsize* $= 128$ bits



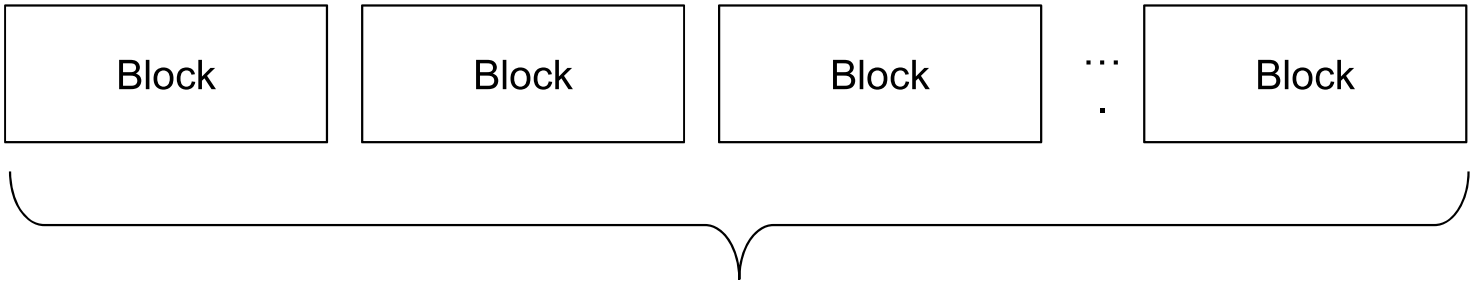
Miniblock

- The miniblock is an atomic unit of information
- It is a portion of a block
- Its size (*msize* bits) is a divisor of the size of the block
- With 4 miniblocks in an AES block $\rightarrow$ *msize* $= 32$ bits



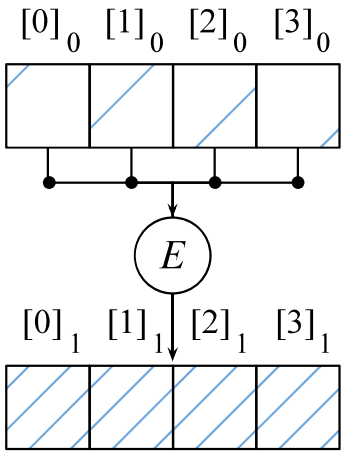
Macroblock

- The macroblock is a sequence of blocks
- Its size *Msize* must be the product of the size of a block *bsize* and a power of the number of mini-blocks in a block
- Macro-block size (for AES and 32-bit miniblocks):
  - $128 \times (128/32)^i =$
    128×4^i bits $=$
    16×4^i bytes $(16, 64, 256, 1024, 4096, \ldots$ bytes$)$



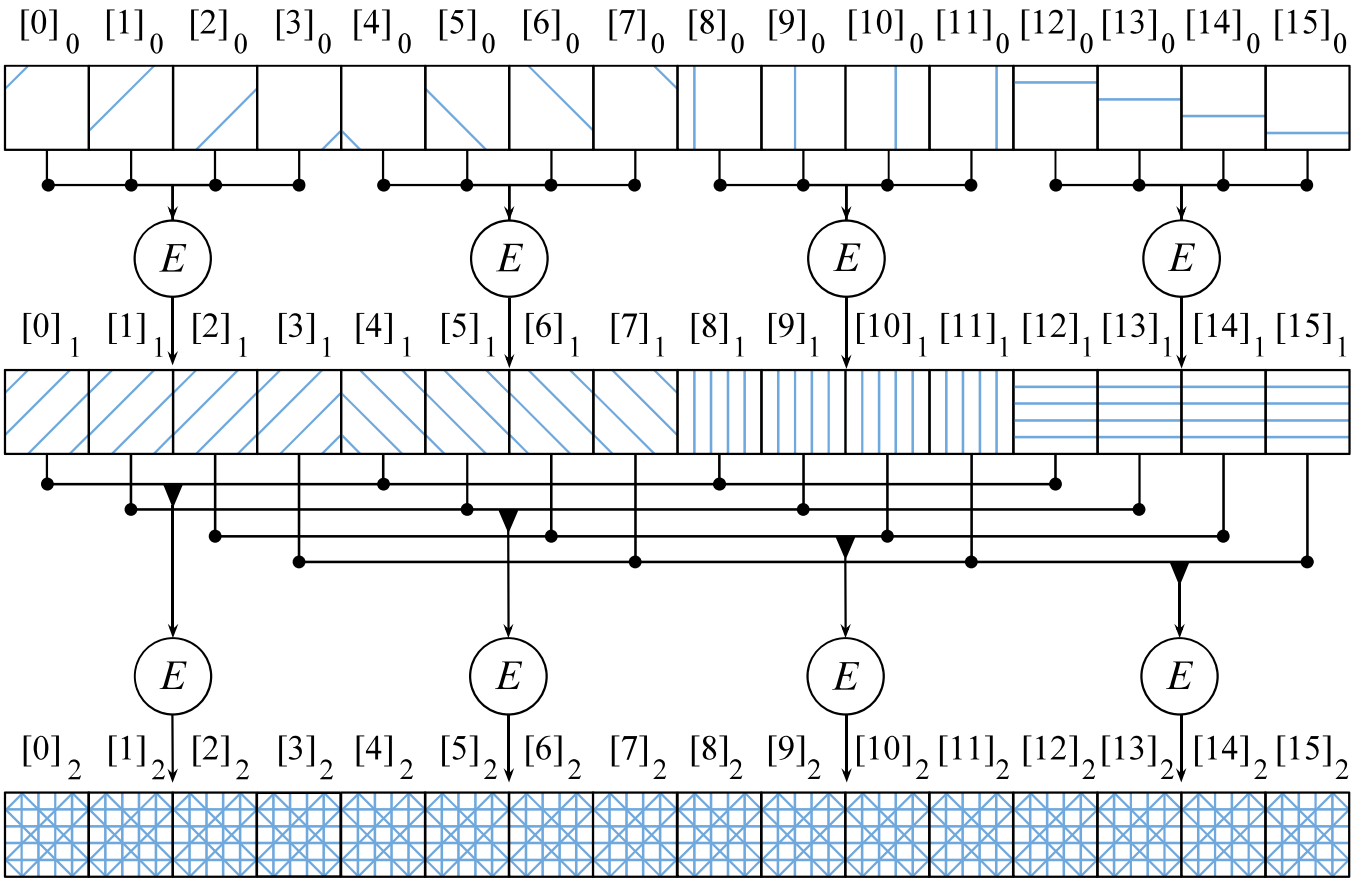
Mixing of a single block

- When encryption is applied to a block, all the miniblocks are mixed



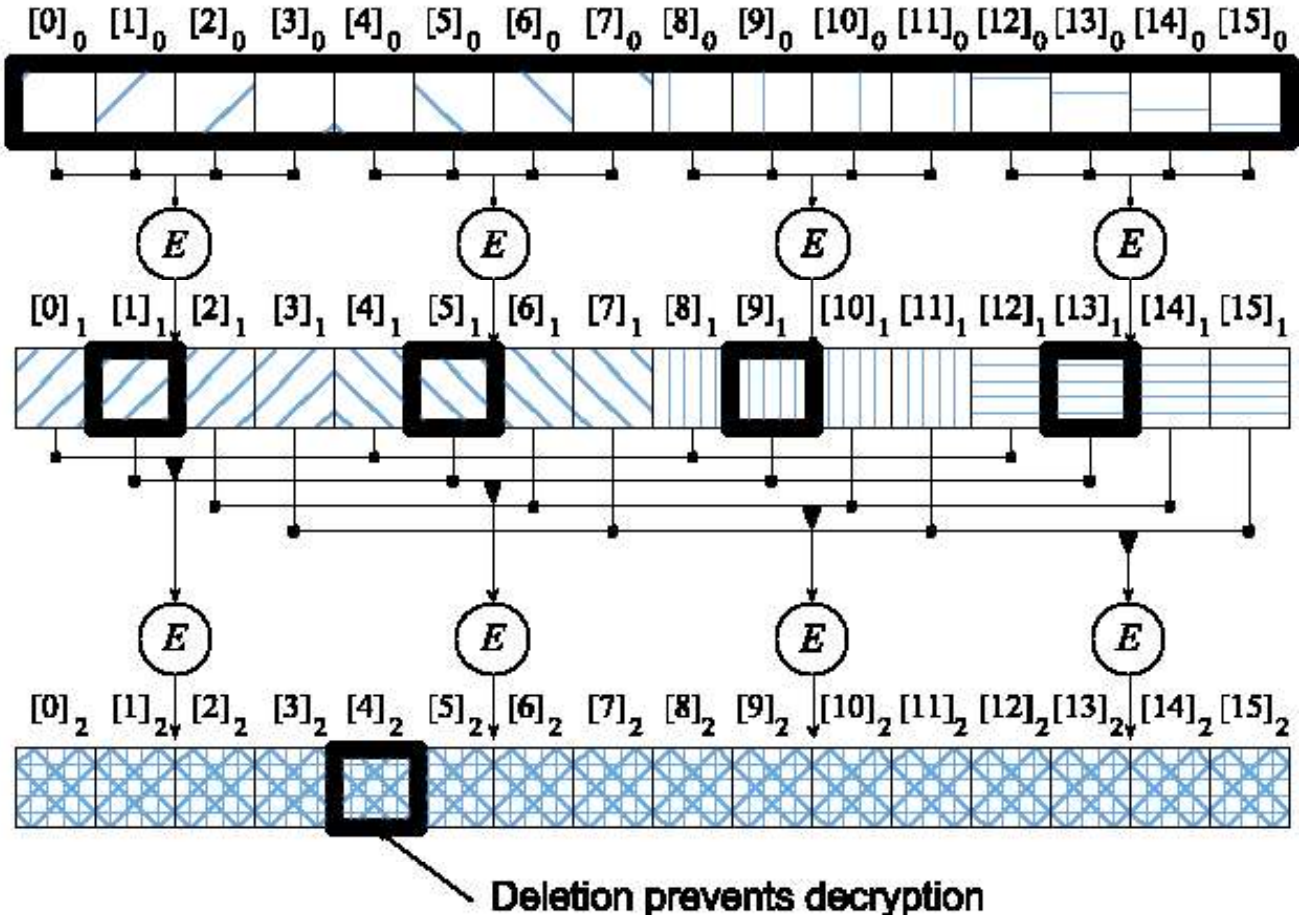
Mixing at 2 levels

- The mixing can be iterated a second time



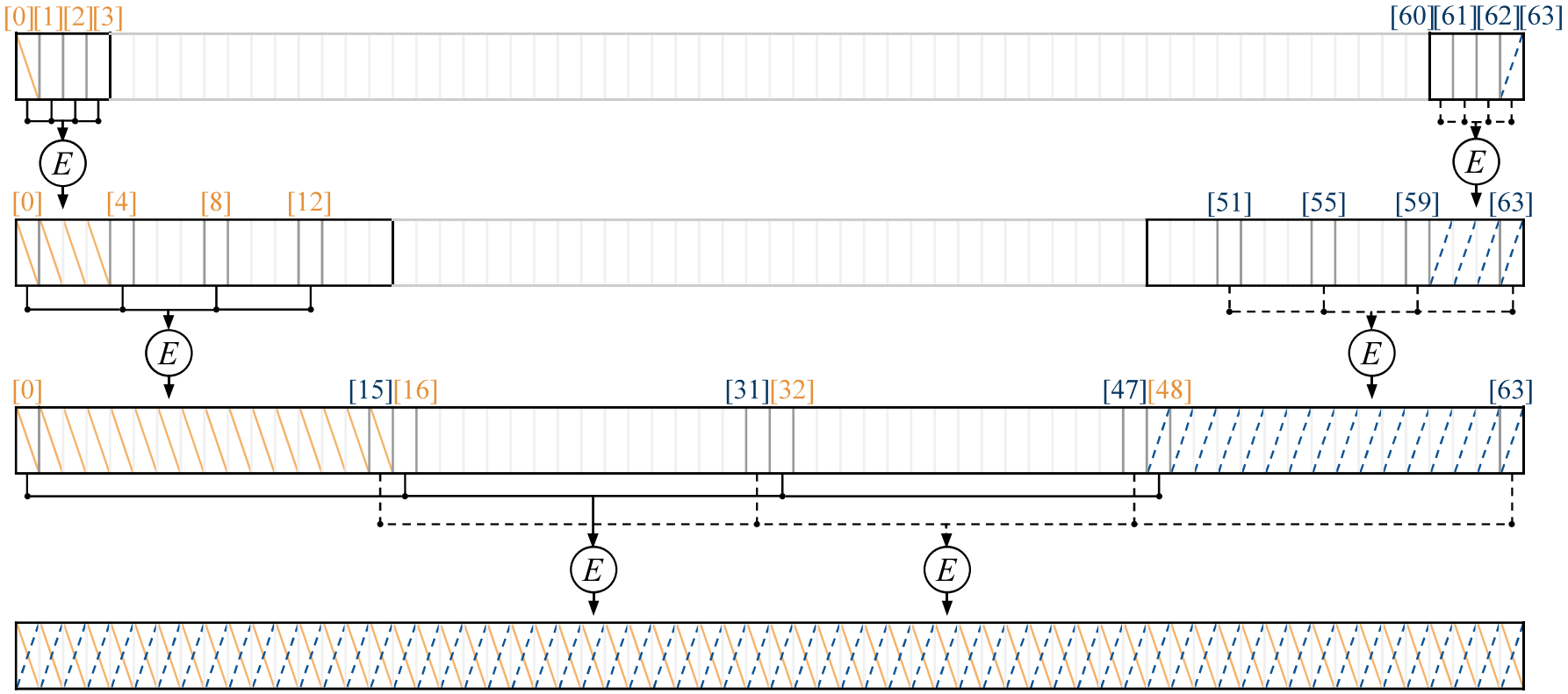
Deletion of a miniblock

- If a miniblock in the encrypted representation is missing, the encryption cannot be inverted, even with knowledge of the key
- Each miniblock in the macroblock is needed to obtain the plaintext



Mixing

- The approach can manage arbitrarily large macroblocks
- Round n mixes *bsize*/*msize* miniblocks at distance $(bsize/msize)^{n-1}$, with an exponential increase in *Msize*



round 1
span: 4
distance: 1

round 2
span: 16
distance: 4

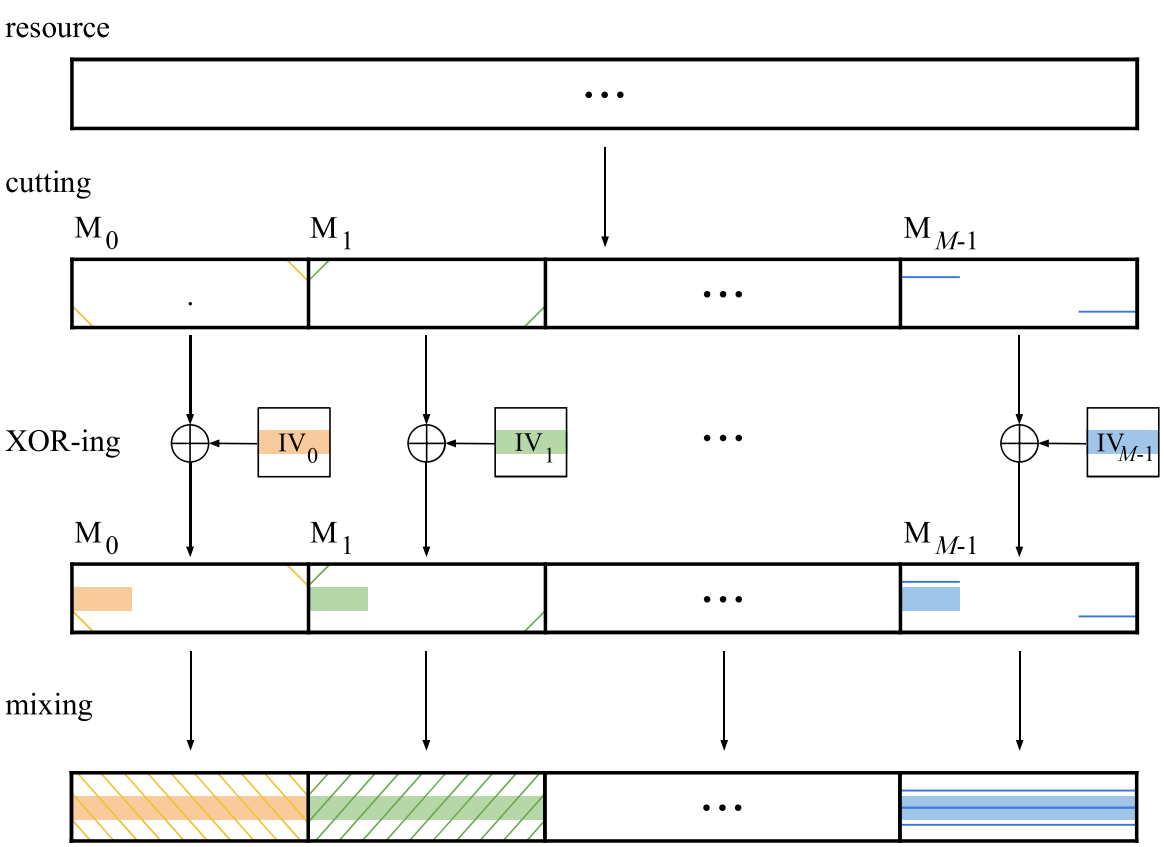
round 3
span: 64
distance: 16

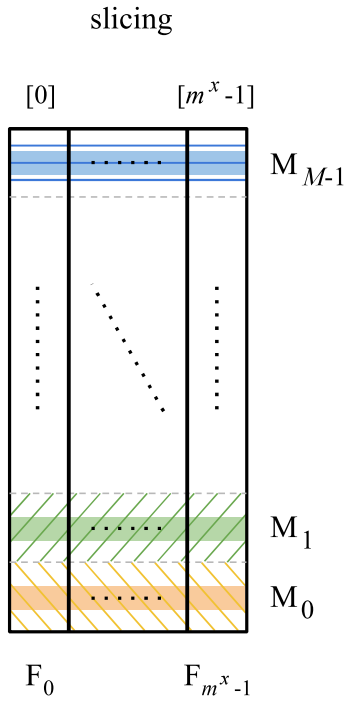
Motivation for using multiple macroblocks

- Excessively large macro-blocks introduce shortcomings:
  - Reduction in decryption efficiency
    - The number of rounds can become large
  - Computational and data transfer overhead
    - When only a small portion of the macroblock is needed, the whole macroblock must be downloaded
  - Increased latency
    - Before plaintext bits are obtained, the whole macroblock must be decrypted
- All these problems can be mitigated by limiting the size of the macro-block

Slicing

- The resource is split into a set of macro-blocks
- All the mini-blocks in the same position will build a slice, called fragment
- Each fragment will be necessary to reconstruct the resource



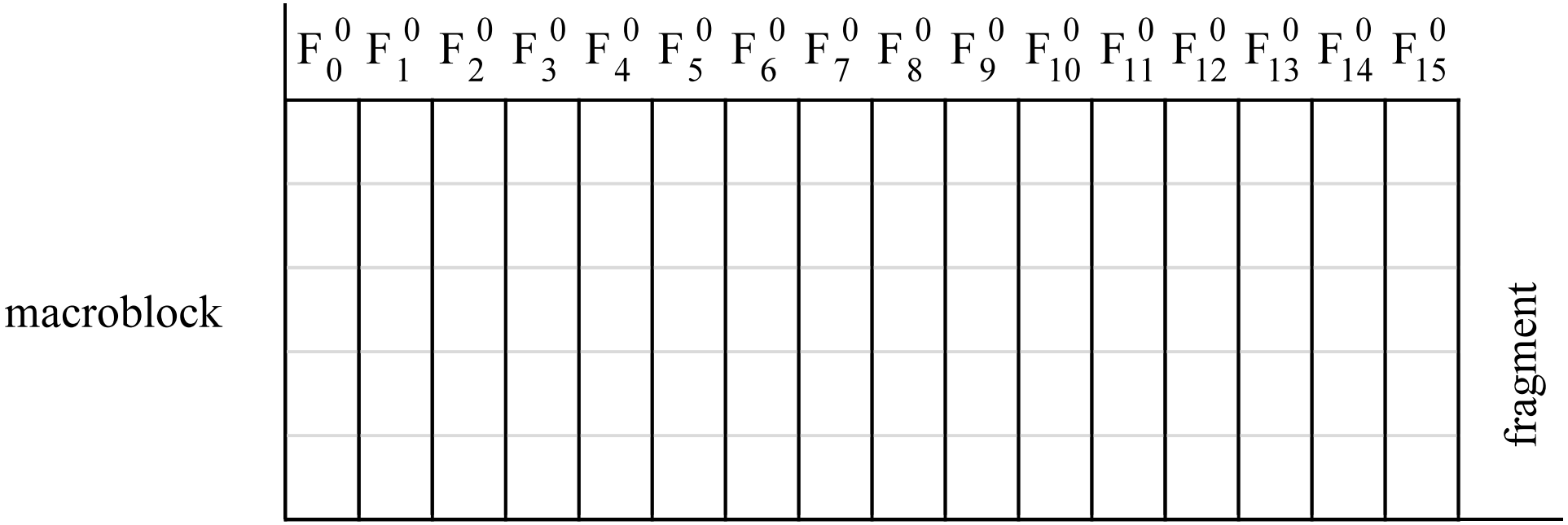


Policy updates

- When a user is revoked access to the object, a randomly chosen fragment is updated
- The start is the base version of F_0^i produced by the application of Mix&Slice
- Fragment F_0^i is replaced at step j by a fragment $F_j^i = E_{k_j}(F_0^i)$
- Authorized users will be able to access k_j
- Revoked users are assumed to still have access to the last key $k_i, i < j$ valid when they last had access to the resource
- Keys can be managed with key regression, based on RSA
  - All keys k_i can be derived in a simple way from $k_j, \forall i < j$, with no need to store additional information

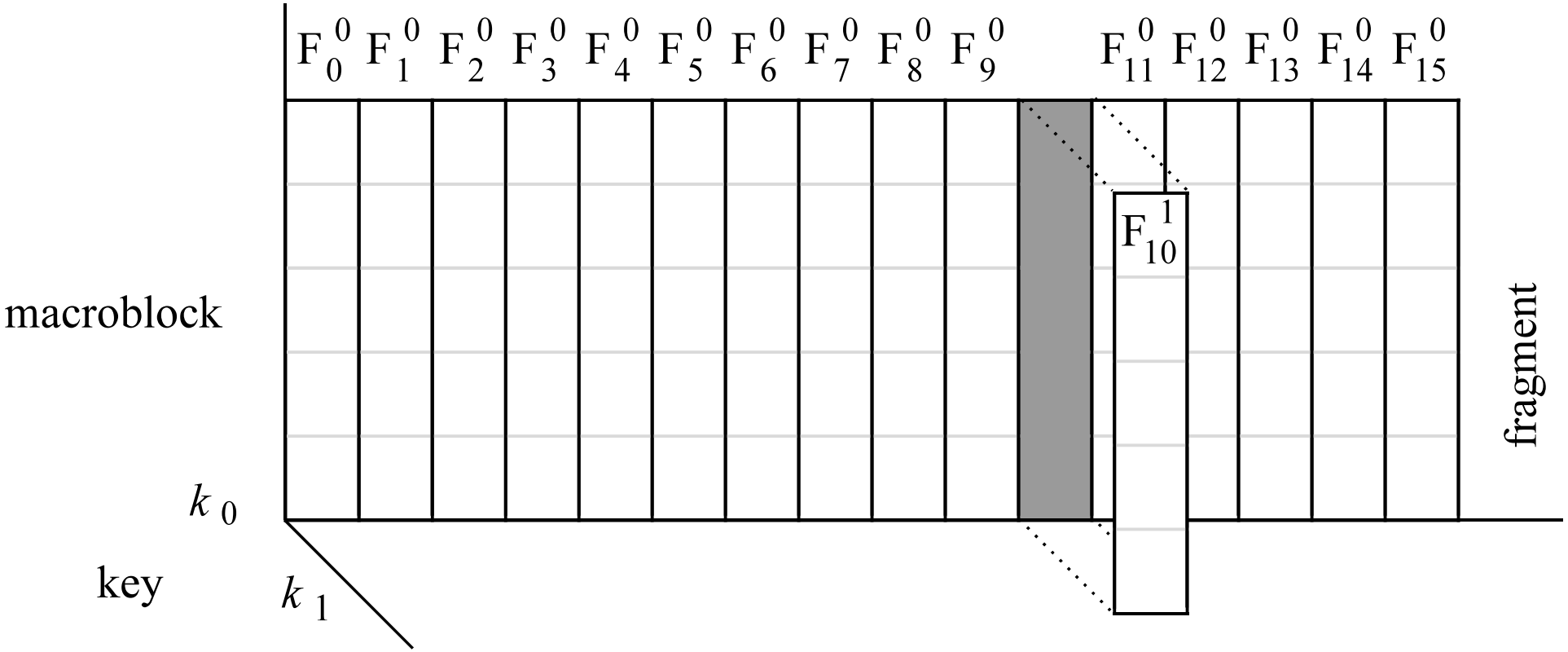
Fragment evolution

- Starting configuration



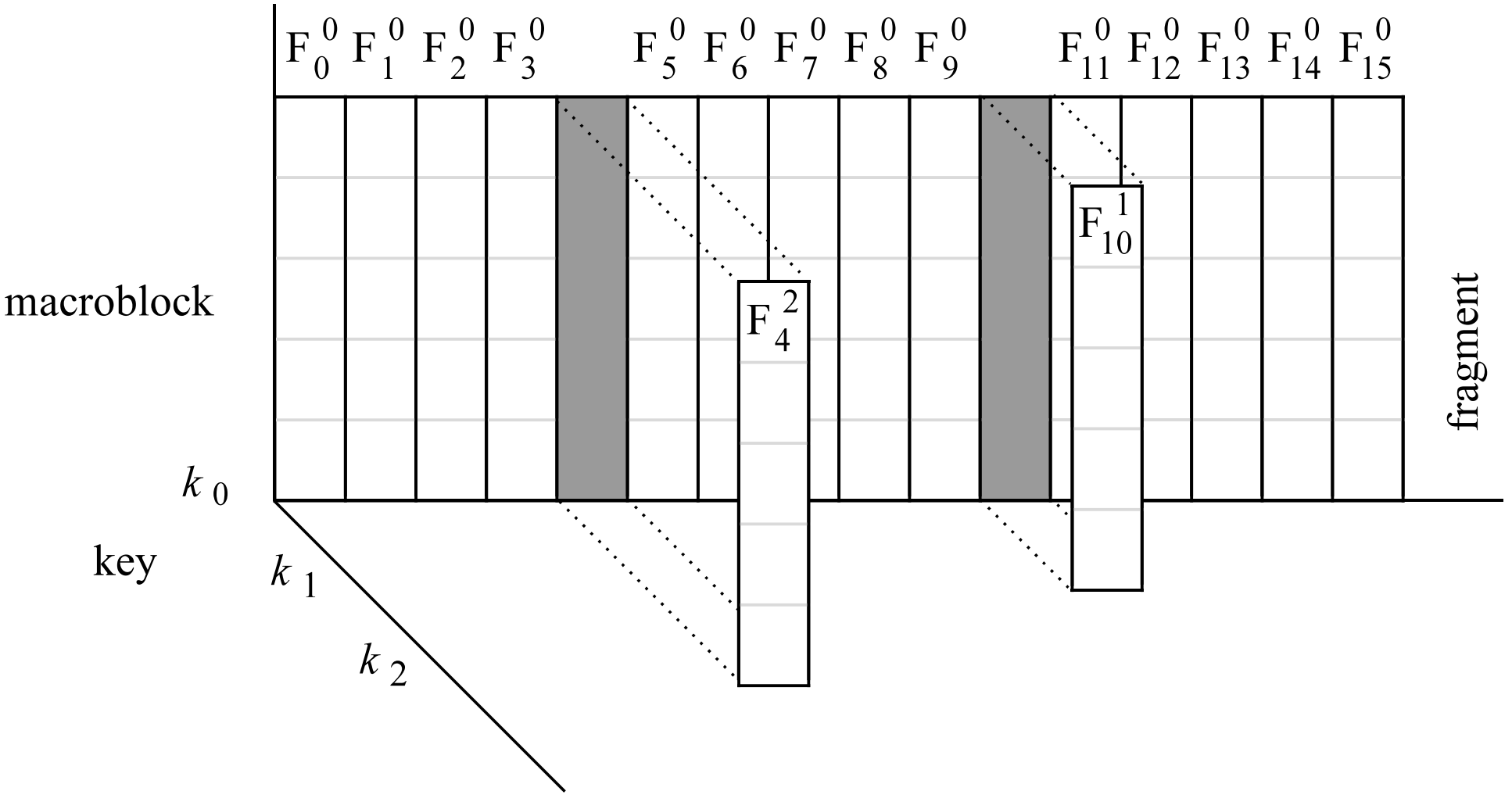
Fragment evolution

- First revocation; F_{10}^{0} is replaced by $\mathrm{F}_{10}^{1} = E_{k_1}(\mathrm{F}_{10}^{0})$



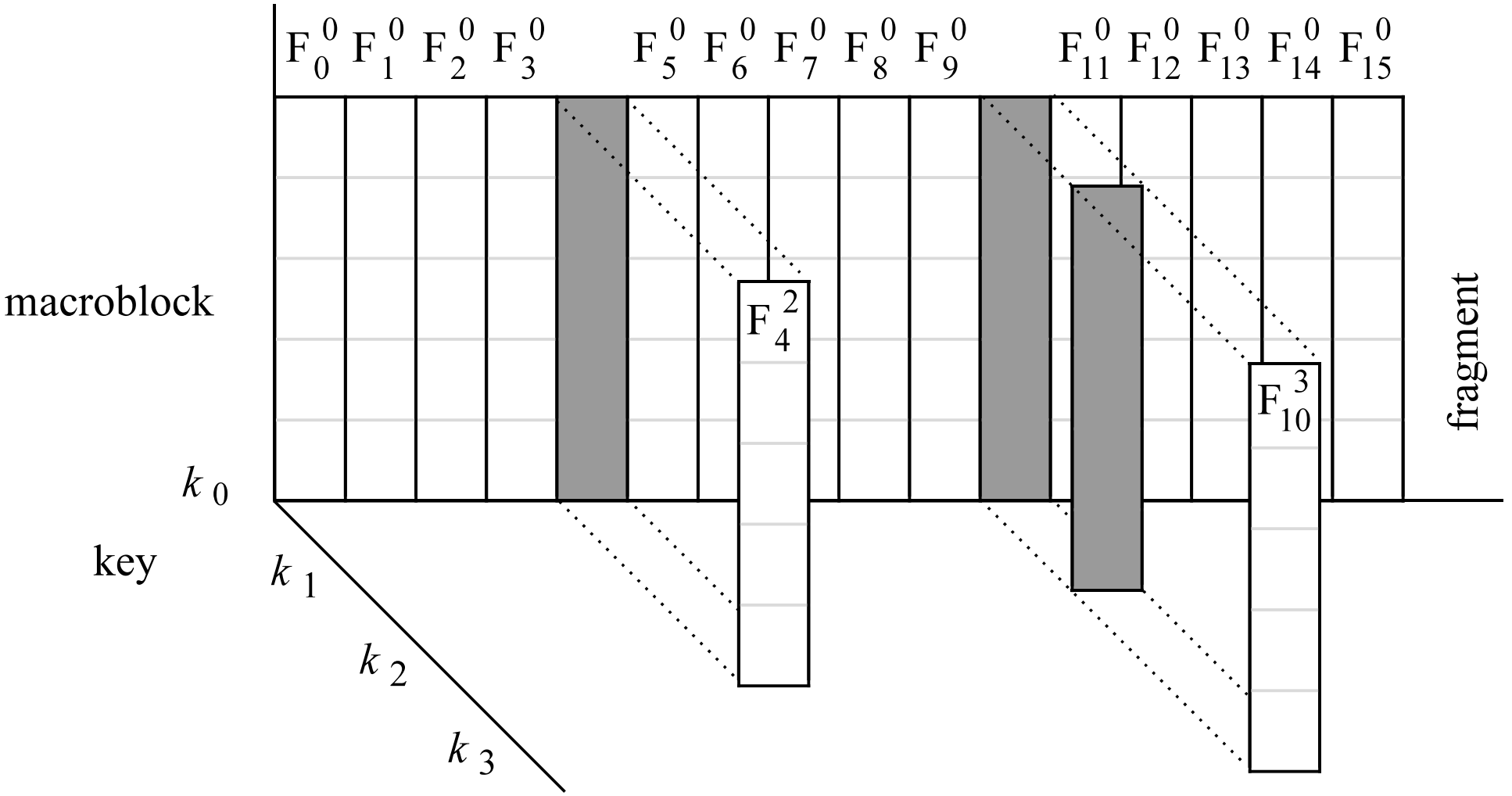
Fragment evolution

- Second revocation; F_4^0 is replaced by $\mathrm{F}_4^2 = E_{k_2}(\mathrm{F}_4^0)$



Fragment evolution

- Third revocation; F_{10}^{1} is replaced by $\mathrm{F}_{10}^{3} = E_{k_3}(\mathrm{F}_{10}^{0})$



Security profile of the domain

- The revoked user can copy the (decrypted) resource to another location (external), when she is still allowed
  - Impossible to mitigate
  - But, resources can be extremely large (e.g., multimedia collections)
    - High storage and transfer costs
- The revoked user must not be able to bypass the protection with a computational and local storage cost significantly lower than the cost of the storage of the resource itself
  - Several AONT techniques do not satisfy this requirement

Possible attacks by the revoked user (1)

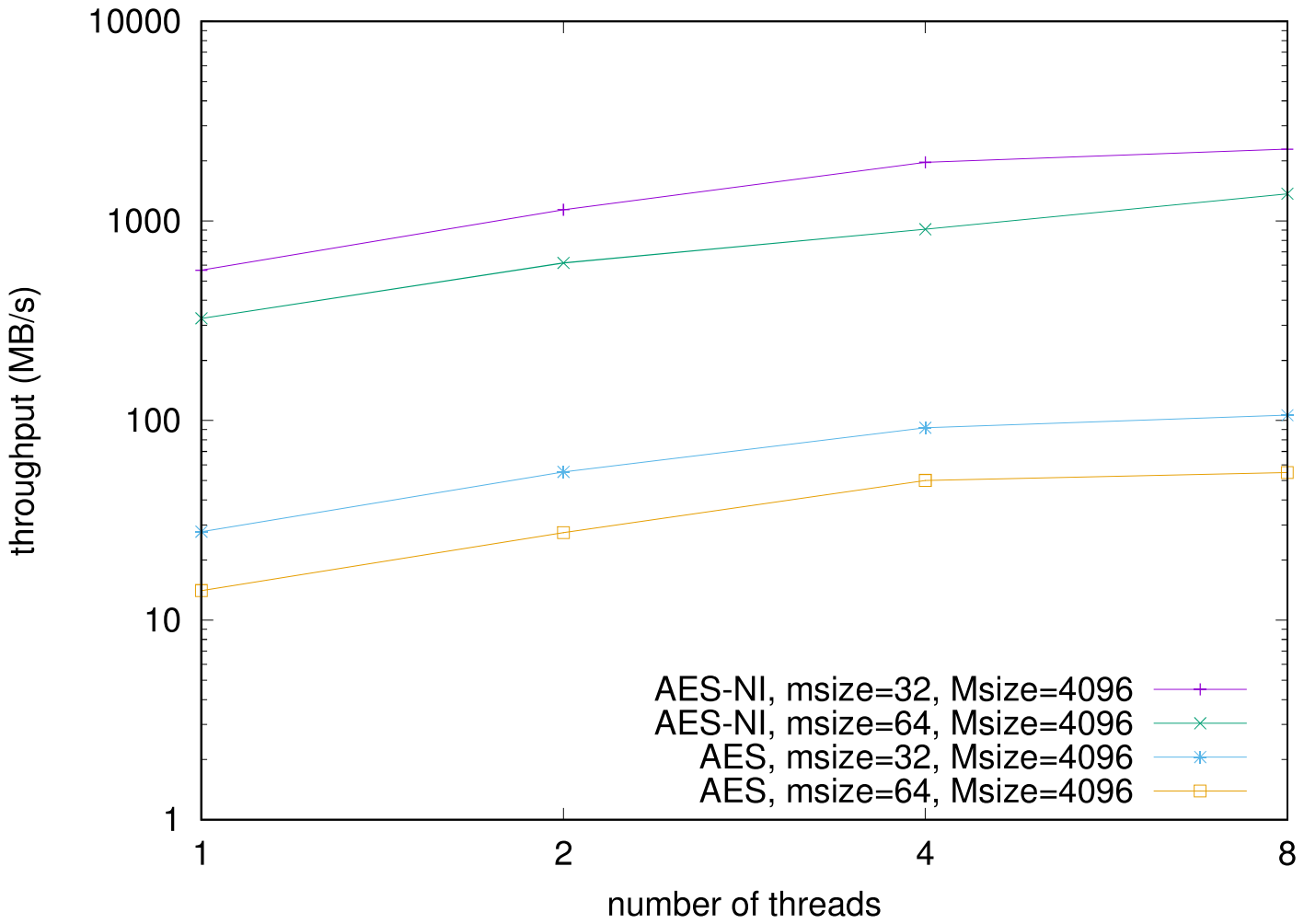
- Store externally a subset of the fragments at level 0
  - Effectiveness after a single change proportional to the number of fragments
    - E.g., storing half the fragments, there is a $p = 0.5$ chance of hitting the fragment that was updated
  - As policy changes accumulate, the effectiveness sees an exponential decrease
    - E.g., storing half the fragments, after 4 updates there is a $p = 2^{-4}$ chance of hitting all the updated fragments

Possible attacks by the revoked user (2)

- Store externally a portion of each fragment/miniblock at level 0
  - The approach trades-off storage cost for a reduction in the cost of attacking the cryptographic protection
  - As policy changes accumulate, the removal of mini-blocks/fragments from the same level 0 block leads to a sudden increase in the computational cost
  - This must be considered when choosing the size *msize* of the mini-block

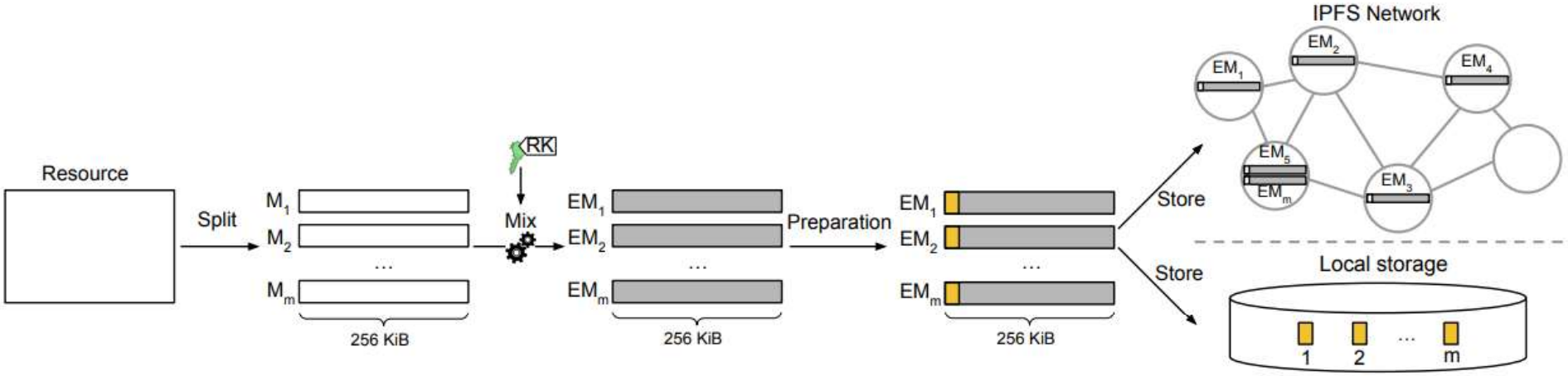
Encryption performance on the client

- Excellent performance, up to 2.5GB/s



Mixing applied in IPFS networks

- The approach can be applied in IPFS
- A golden fragment is kept locally
- Protection is independent from the ability to break cryptographic protection



M. Abbadini, M. Beretta, S. De Capitani di Vimercati, D. Facchinetti, S. Foresti, G. Oldani, S. Paraboschi, M. Rossi and P. Samarati, "Supporting Data Owner Control in IPFS Networks," in Proc. of 2024 IEEE Int. Conf. on Communications (ICC 2024)

Computation integrity – 1

- Data processing may be performed by non trustworthy providers
- Need mechanisms that provide integrity for computation results:
  - correctness: computed on genuine data
  - completeness: computed on the whole data collection
  - freshness: computed on the most recent version of the data

[1]

[1] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, R. Sassi, P. Samarati, "Sentinels and Twins: Effective Integrity Assessment for Distributed Computation," in *IEEE TPDS*, vol. 34, n. 1, January 2023.

S. De Capitani di Vimercati, S. Foresti, S. Paraboschi, P. Samarati, "Data Privacy and Computation Integrity in Machine Learning Scenarios: Some Issues and Approaches," in *Bulletin of the IEEE TCDE*, December 2025.

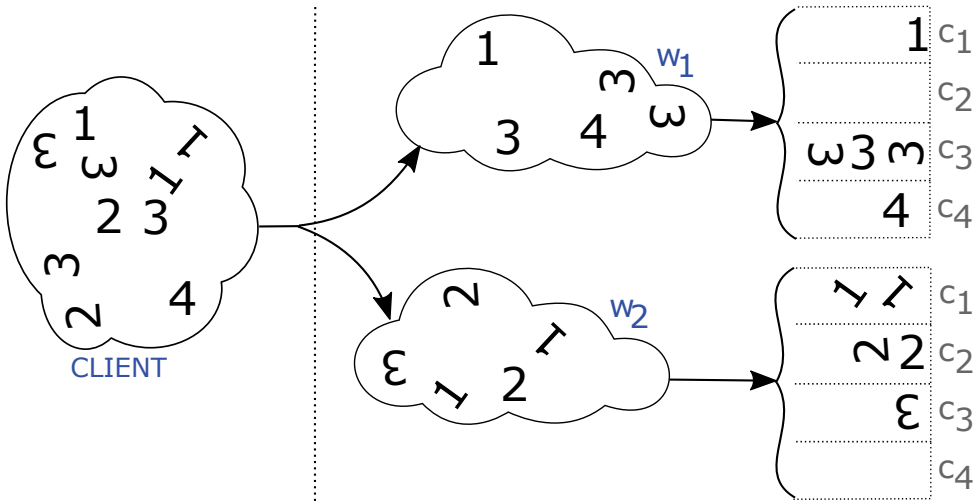
Computation integrity – 2

- Deterministic solutions based on a data structure (e.g., signature chains, Merkle hash trees, skip lists), need knowledge of the workload

- Probabilistic solutions based on dynamic insertion of control information:
  - sentinels: fake tasks for which result is known
  - data job replication: replicated tasks to check consistency in the result

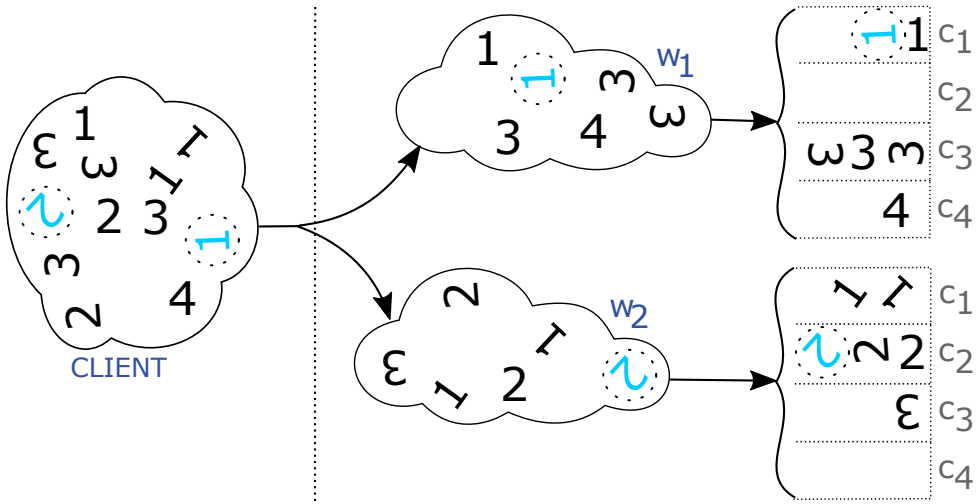
Integrity checks

- Integrity control for distributed ML (e.g., classification) jobs
- Distribute job execution (e.g., classification tasks) to workers



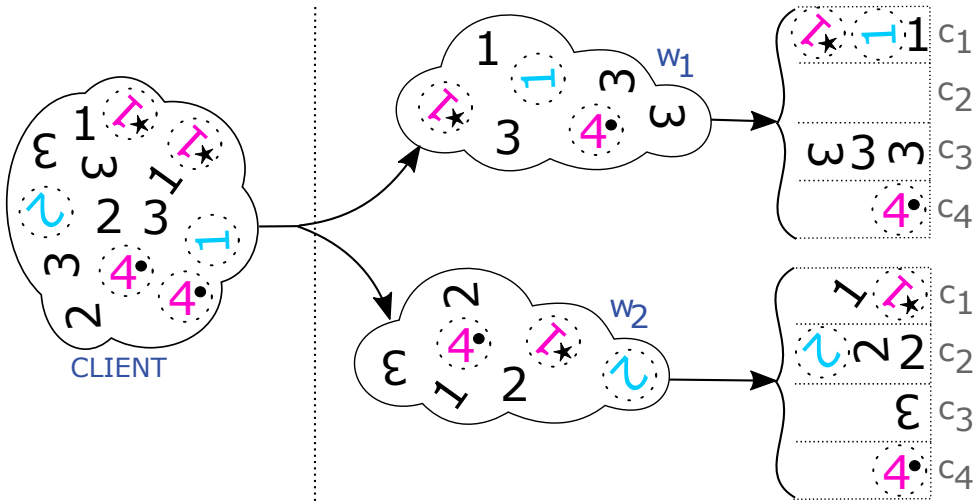
Integrity checks

- Integrity control for distributed ML (e.g., classification) jobs
- Distribute job execution (e.g., classification tasks) to workers
- Inject pre-computed jobs (sentinels)



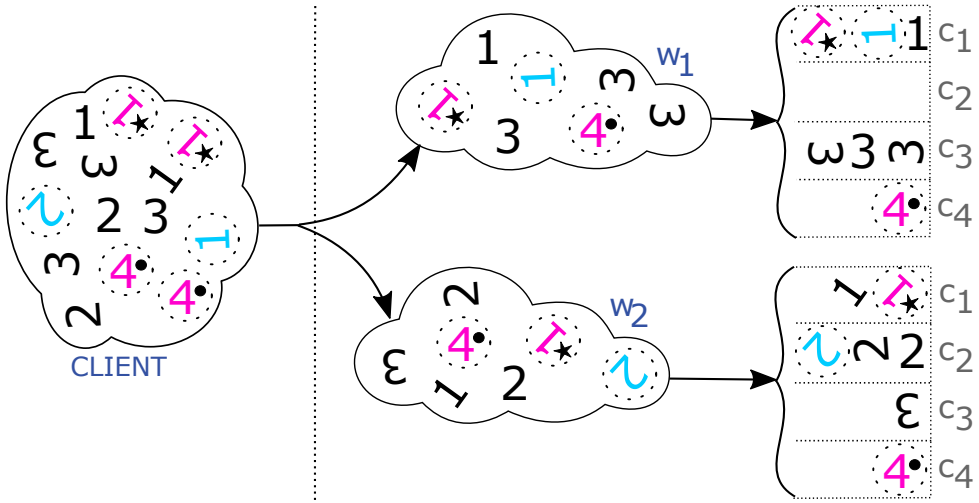
Integrity checks

- Integrity control for distributed ML (e.g., classification) jobs
- Distribute job execution (e.g., classification tasks) to workers
- Inject pre-computed jobs (sentinels) and replicated jobs (twins)



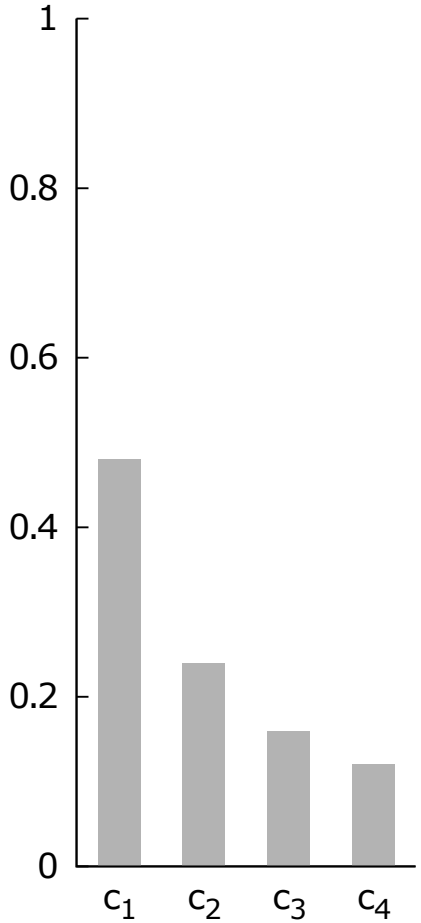
Integrity checks

- Integrity control for distributed ML (e.g., classification) jobs
- Distribute job execution (e.g., classification tasks) to workers
- Inject pre-computed jobs (sentinels) and replicated jobs (twins)
  - how should sentinels be distributed in the class?
  - better more twin pairs or more replicas of the same job?
  - how many sentinels and twins for achieving a given guarantee?



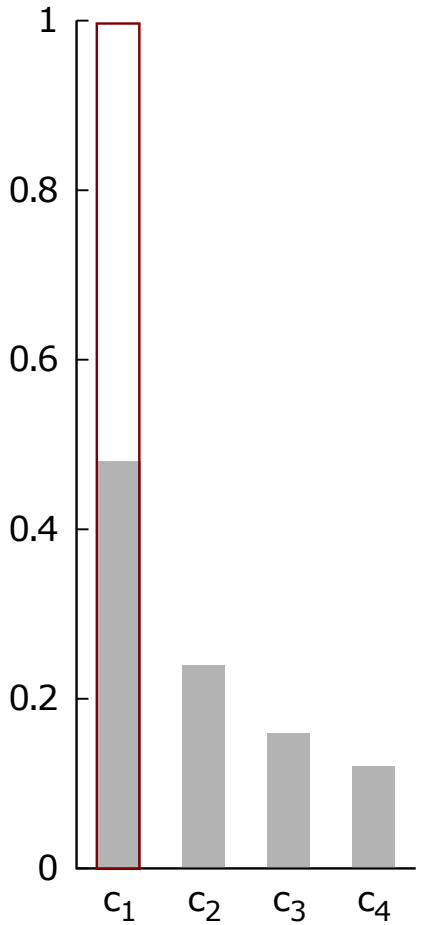
Sentinels

- Signal violation if job is omitted AND the returned result is wrong



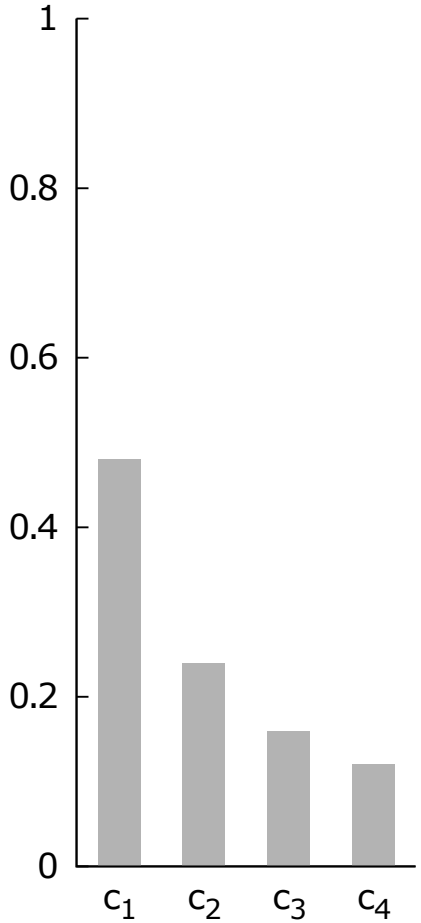
Sentinels

- Signal violation if job is omitted AND the returned result is wrong
- Lazy worker most convenient guessing strategy:
  - always return one class
    ideally, the most frequent for sentinels



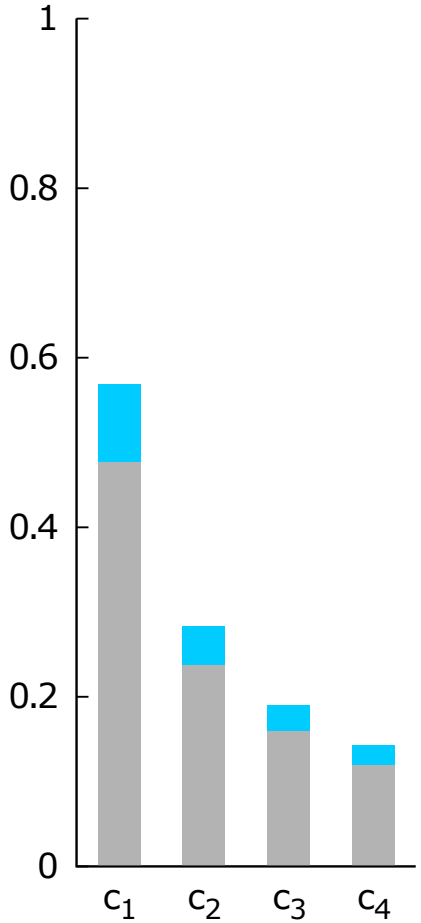
Sentinels

- Signal violation if job is omitted AND the returned result is wrong
- Lazy worker most convenient guessing strategy:
  - always return one class
    ideally, the most frequent for sentinels
- Which client distribution strategy for sentinels into classes is better?
  - genuine data distribution
  - inverse of genuine data distribution
  - uniform distribution



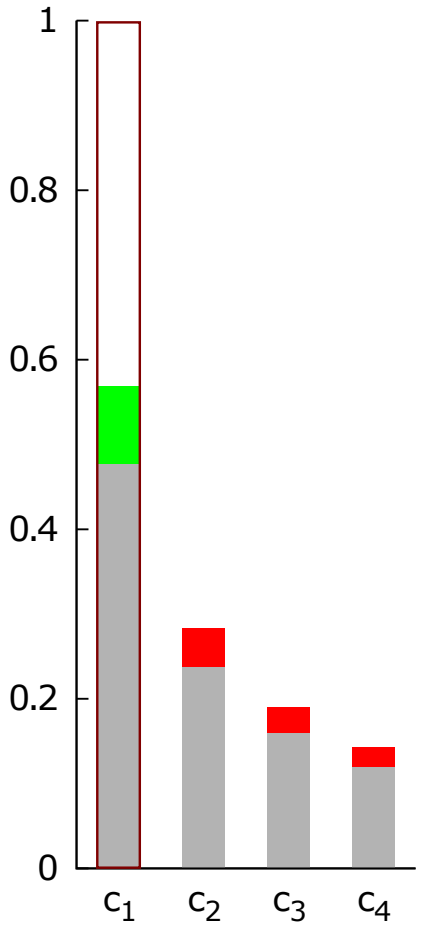
Sentinels

- Signal violation if job is omitted AND the returned result is wrong
- Lazy worker most convenient guessing strategy:
  - always return one class
    ideally, the most frequent for sentinels
- Which client distribution strategy for sentinels into classes is better?
  - genuine data distribution
  - inverse of genuine data distribution
  - uniform distribution



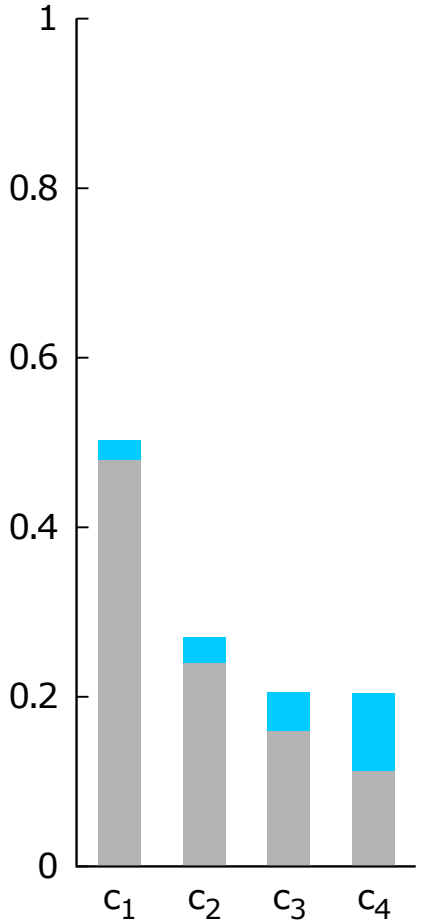
Sentinels

- Signal violation if job is omitted AND the returned result is wrong
- Lazy worker most convenient guessing strategy:
  - always return one class
    ideally, the most frequent for sentinels
- Which client distribution strategy for sentinels into classes is better?
  - genuine data distribution
  - inverse of genuine data distribution
  - uniform distribution



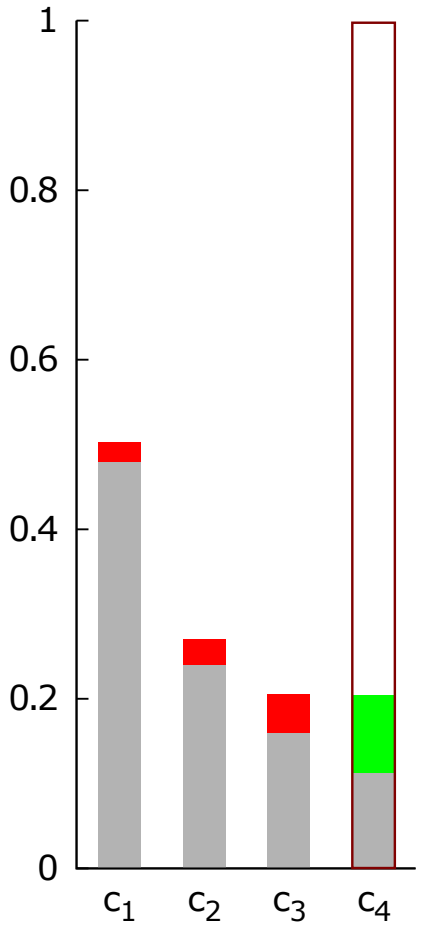
Sentinels

- Signal violation if job is omitted AND the returned result is wrong
- Lazy worker most convenient guessing strategy:
  - always return one class
    ideally, the most frequent for sentinels
- Which client distribution strategy for sentinels into classes is better?
  - genuine data distribution
  - inverse of genuine data distribution
  - uniform distribution



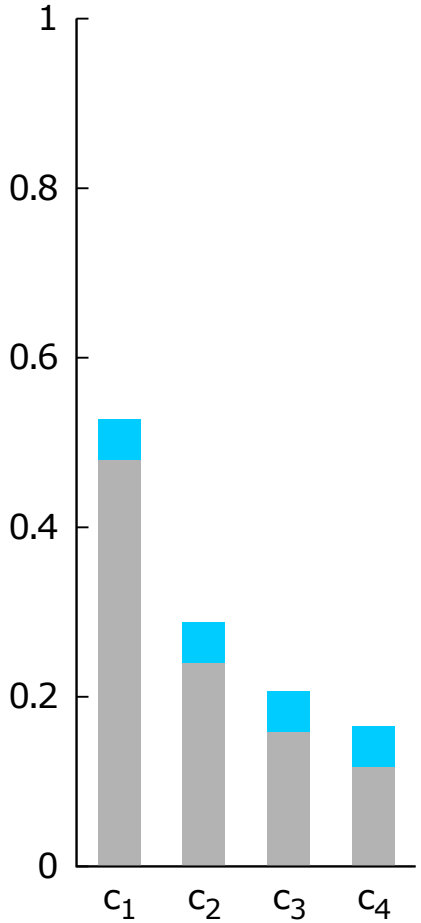
Sentinels

- Signal violation if job is omitted AND the returned result is wrong
- Lazy worker most convenient guessing strategy:
  - always return one class
    ideally, the most frequent for sentinels
- Which client distribution strategy for sentinels into classes is better?
  - genuine data distribution
  - inverse of genuine data distribution
  - uniform distribution



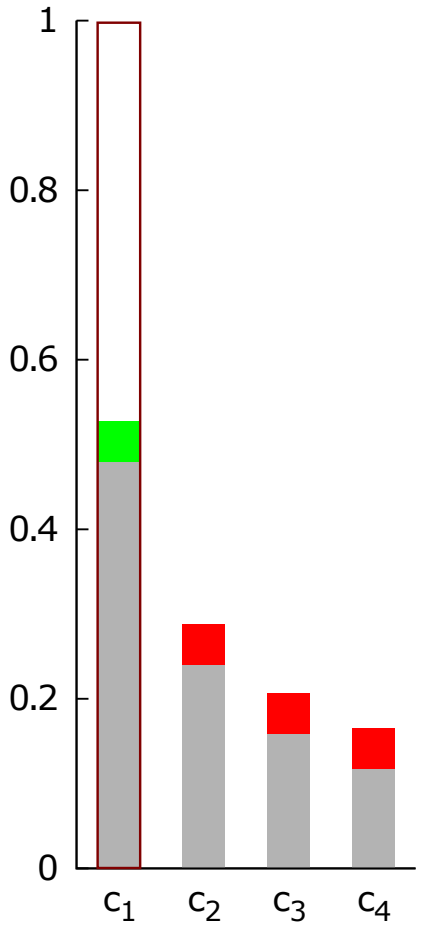
Sentinels

- Signal violation if job is omitted AND the returned result is wrong
- Lazy worker most convenient guessing strategy:
  - always return one class
    ideally, the most frequent for sentinels
- Which client distribution strategy for sentinels into classes is better?
  - genuine data distribution
  - inverse of genuine data distribution
  - uniform distribution



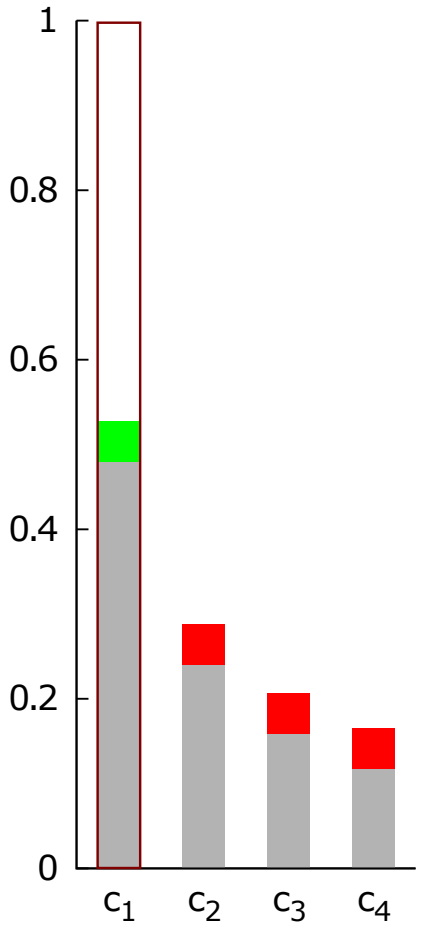
Sentinels

- Signal violation if job is omitted AND the returned result is wrong
- Lazy worker most convenient guessing strategy:
  - always return one class
    ideally, the most frequent for sentinels
- Which client distribution strategy for sentinels into classes is better?
  - genuine data distribution
  - inverse of genuine data distribution
  - uniform distribution



Sentinels

- Signal violation if job is omitted AND the returned result is wrong
- Lazy worker most convenient guessing strategy:
  - always return one class
    ideally, the most frequent for sentinels
- Which client distribution strategy for sentinels into classes is better?
  - genuine data distribution
  - inverse of genuine data distribution
  - uniform distribution
- Better use uniform distribution

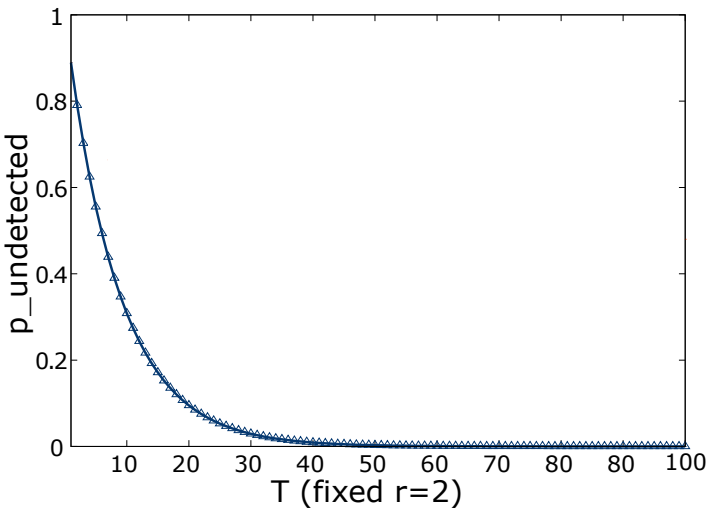


Twins

- Signals violation if results are in disagreement
- Better to use more twin sets or a higher replication factor?

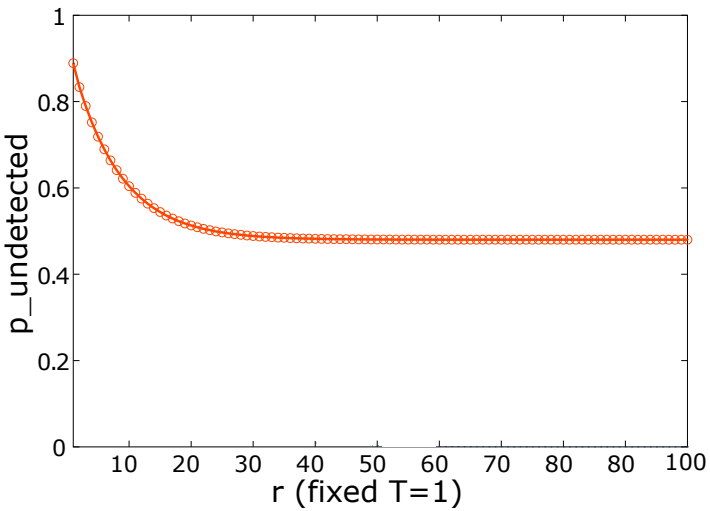
Twins

- Signals violation if results are in disagreement
- Better to use more twin sets or a higher replication factor?
- The probability of an omission going undetected decreases:
  - exponentially with twin sets
  - very slowly with replication factor



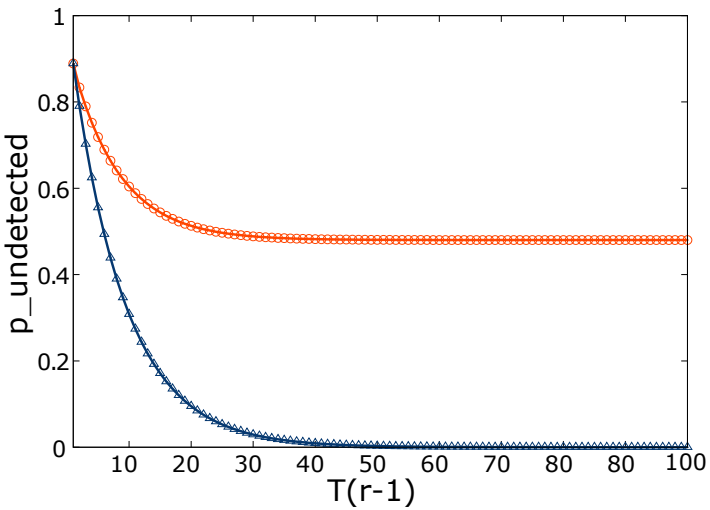
Twins

- Signals violation if results are in disagreement
- Better to use more twin sets or a higher replication factor?
- The probability of an omission going undetected decreases:
  - exponentially with twin sets
  - very slowly with replication factor



Twins

- Signals violation if results are in disagreement
- Better to use more twin sets or a higher replication factor?
- The probability of an omission going undetected decreases:
  - exponentially with twin sets
  - very slowly with replication factor



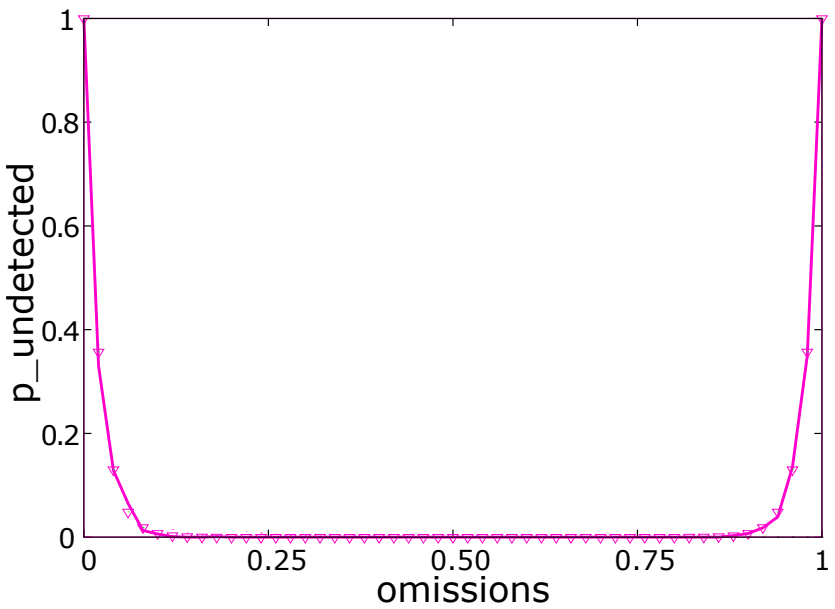
- Better use one replica and a higher number of twin sets

Effectiveness of twins

- Twins are more (roughly twice) effective than sentinels
- Twins lose their effectiveness with:
  - highly skewed data distribution: high probability of correct guess

Effectiveness of twins

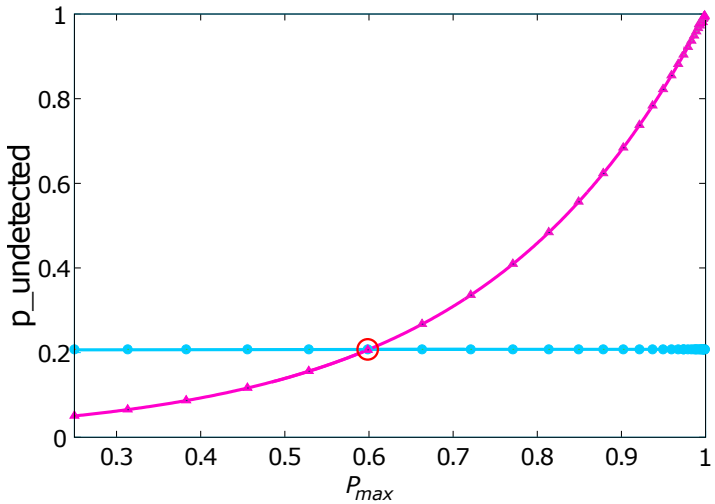
- Twins are more (roughly twice) effective than sentinels
- Twins lose their effectiveness with:
  - highly skewed data distribution: high probability of correct guess
  - extreme omissions: high probability of omitting pairs of twins



Combining sentinels and twins

- The number of sentinels/twin sets depends on the threshold on the probability of omissions going undetected
  - the lower the probability, the higher the number of control jobs

- Twins or sentinels?
  - sentinels if $P_{max} > \frac{1}{2} \cdot (1 + \frac{1}{c})$
  - twins (and a handful of sentinels to detect extreme omissions) otherwise

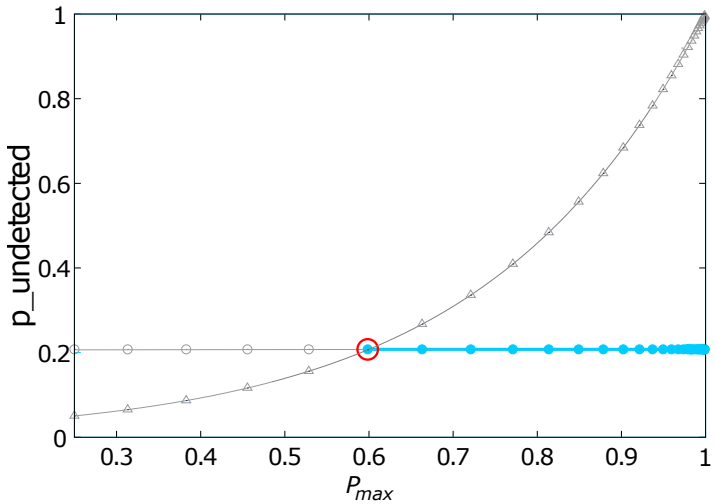


P_{max}: probability of the most frequent class
c: number of classes

Combining sentinels and twins

- The number of sentinels/twin sets depends on the threshold on the probability of omissions going undetected
  - the lower the probability, the higher the number of control jobs

- Twins or sentinels?
  - sentinels if $P_{max} > \frac{1}{2} \cdot (1 + \frac{1}{c})$
  - twins (and a handful of sentinels to detect extreme omissions) otherwise

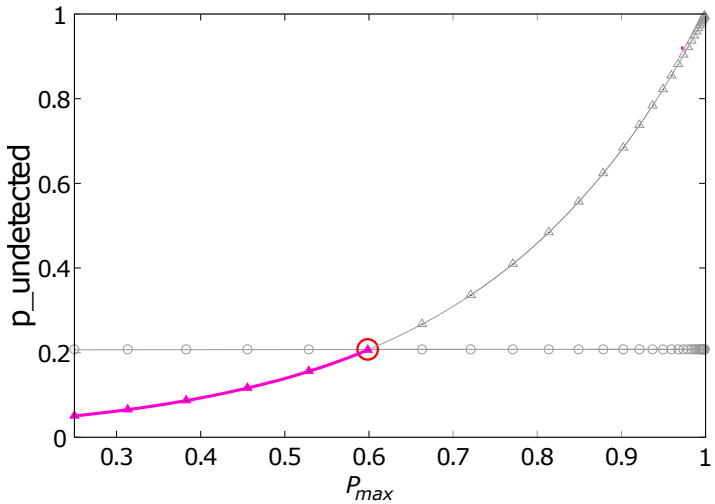


P_{max}: probability of the most frequent class
c: number of classes

Combining sentinels and twins

- The number of sentinels/twin sets depends on the threshold on the probability of omissions going undetected
  - the lower the probability, the higher the number of control jobs

- Twins or sentinels?
  - sentinels if $P_{max} > \frac{1}{2} \cdot (1 + \frac{1}{c})$
  - twins (and a handful of sentinels to detect extreme omissions) otherwise

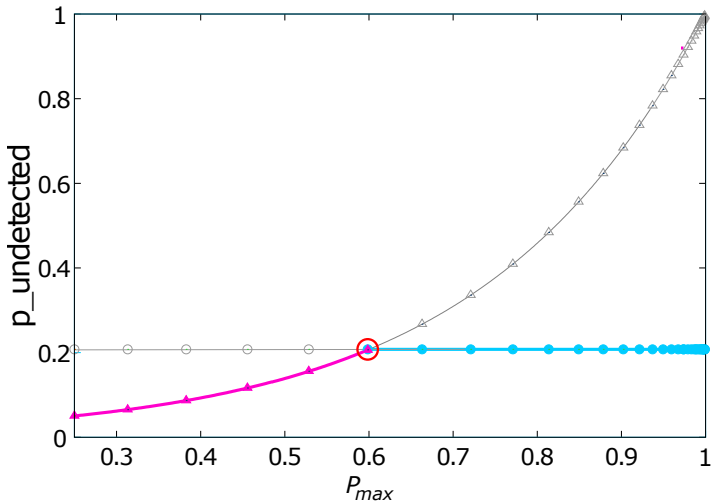


P_{max}: probability of the most frequent class
c: number of classes

Combining sentinels and twins

- The number of sentinels/twin sets depends on the threshold on the probability of omissions going undetected
  - the lower the probability, the higher the number of control jobs

- Twins or sentinels?
  - sentinels if $P_{max} > \frac{1}{2} \cdot (1 + \frac{1}{c})$
  - twins (and a handful of sentinels to detect extreme omissions) otherwise



P_{max}: probability of the most frequent class
c: number of classes

Open issues

- Support for different ML/AI tasks
- Support for non-deterministic tasks
- Collusion among workers
- Identification of lazy/malicious workers
- Different trust assumptions, possibly evolving over time, on workers
- Generation of unrecognizable sentinels

AI/ML for data and application security

- AI/ML is (obviously) having a great impact on every domain
  - It also holds for this research area
- A few consequences
  - AI/ML can support attackers, but defenders may gain more
  - AI/ML tools can lead to more flexible tools, for policy specification and enforcement
  - Research becomes more empirical
  - Availability of artifacts and verifiation will become more common
  - The amount of work associated with a major publication (top conference) increases

Concluding remarks

- The techniques presented illustrate some of the many aspects that have to be considered when dealing with data protection
- Overall goal of data protection
  - Giving control to data owners
- This needs models and techniques, as the basis for the realization of future technology